Project Title: IoT Flood Monitoring System

Components Required:

Hardware:

Arduino or Raspberry Pi (as the main controller)
Water level sensors (ultrasonic, capacitive, or float sensors)
IoT communication module (e.g., ESP8266 or SIM800)
Power source (battery or solar panel)
Enclosure for electronics (to protect them from the elements)
Software:

Arduino IDE or Python (for programming the controller)
IoT platform (such as AWS IoT, Google Cloud IoT, or ThingSpeak)
Mobile app or web interface for data visualization
Project Steps:

Sensor Setup:

Connect the water level sensors to the Arduino or Raspberry Pi.
Calibrate the sensors to accurately measure water levels.
Place the sensors at strategic locations prone to flooding (near rivers, lakes, or flood-prone areas).
Controller Programming:

Write code to read data from the sensors.
Implement a threshold system to detect rising water levels.
Integrate the IoT communication module to send data to the cloud.
Cloud Integration:

Set up an IoT platform account (e.g., AWS IoT, Google Cloud IoT).
Create a Thing or Device to receive data from the controller.
Configure security and authentication for data transmission.
Data Visualization:

Develop a mobile app or web interface to display real-time water level data.
Add features like alerts and historical data graphs for better monitoring.
Alert System:

Implement an alert system to notify authorities or users when water levels exceed predefined thresholds.
Use email, SMS, or push notifications for alerts.
Power Management:

If using batteries, implement power-saving techniques to extend the device's life.
Consider using a solar panel for sustainable power.
Testing and Deployment:

Test the system in a controlled environment to ensure accuracy.
Deploy the sensors and controllers in flood-prone areas.
Maintenance:

Regularly monitor and maintain the system to ensure it operates correctly.
Replace batteries or perform maintenance on solar panels as needed.
Data Analytics (Optional):

For more advanced projects, you can analyze historical data to predict and model flood patterns.
Remember that this is a basic outline, and you can expand and customize the project based on your specific requirements and available resources. Proper planning, testing, and safety considerations are crucial when dealing with flood monitoring systems to ensure their effectiveness in real-world scenarios.

```
#include <Wire.h>
#include <Adafruit_SSD1306.h>
#include <NewPing.h>

#define TRIGGER_PIN  7
#define ECHO_PIN     6
#define MAX_DISTANCE 200

NewPing sonar(TRIGGER_PIN, ECHO_PIN, MAX_DISTANCE);

#define SCREEN_WIDTH 128
#define SCREEN_HEIGHT 32
#define OLED_RESET    -1
Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, OLED_RESET);

const int threshold = 50; // Define your water level threshold in centimeters

void setup() {
  Serial.begin(9600);
  display.begin(SSD1306_SWITCHCAPVCC, 0x3C);
  display.display();
  delay(1000);
  display.clearDisplay();
}

void loop() {
  delay(1000);
  int distance = sonar.ping_cm();

  if (distance == 0) {
    Serial.println("Error: No ping received");
  } else {
    Serial.print("Distance: ");
    Serial.print(distance);
    Serial.println(" cm");

    display.clearDisplay();
    display.setTextSize(1);
    display.setTextColor(SSD1306_WHITE);
    display.setCursor(0, 0);
    display.print("Water Level:");
    display.setCursor(0, 10);
    display.print(distance);
    display.print(" cm");
    display.display();

    if (distance <= threshold) {
      Serial.println("Flooding detected!");
      // Implement code here to send alerts, e.g., SMS or email
    }
  }
}
```